

### REMARKS

Claims 1-44 are now pending in the application. Claim 26 has been canceled. Claims 8, 27, 37, and 38 have been amended. No new matter has been added. Reconsideration and reexamination are respectfully requested in view of the amendments and the following remarks.

#### **I. The § 103(a) Rejections**

The Examiner rejected claims 1-44 under 35 U.S.C. § 103(a) over Bristor (U.S. 6,018,342).

Claim 1 recites a method implemented in a computer program application that performs operations on documents having states. The method includes “maintaining in a memory a state history of a document for storing document states,” “automatically capturing the state of the document as it exists after [an operation of a predetermined type],” and “adding the captured state to the state history.” Bristor does not disclose or suggest maintaining a history of the states of a document, or capturing the state of a document after an operation of a predetermined type.

Bristor provides a method for automatically organizing user-generated signals so that a user “can quickly and easily regenerate [them] for transmission to a computer process” (col. 1, lines 8-9; col. 5, lines 59-67; col. 6, lines 1-4). In Bristor’s method, user-generated signals — such as Unix shell commands (col. 1, lines 24-34) or Internet URLs (col. 1, lines 40-44) — are converted into data that convey information via an engine to a computer process (col. 1, lines 13-20; Fig. 7 and col. 12, lines 53-65). These “user data” are accumulated in various categories in a history database (col. 6, lines 8-18; Fig. 8 and col. 13, lines 30-37). The user can, thereafter, take advantage of this sorting of user data to find and re-execute a particular user-generated signal (col. 6, lines 29-43; col. 13, lines 10-28).

Bristor’s teachings do not suggest the accumulation of document states for a document. Indeed, the user data accumulated in Bristor’s method reflect the *history of the user* rather than the *history of a document* being manipulated. For example, Bristor teaches the accumulation and sorting of the Unix shell commands entered by a user (col. 8, lines 51-67) or the Internet URLs visited by a user (col. 9, lines 20-30). Such user data may cause a processor to return, for

example, a list of files in a directory or an image of a web page. Bristor's method allows the user to recall a signal generated earlier in the user's history and re-execute it — independently of the state of any documents to which the signals relate.

Thus, Bristor teaches a method for tracing and ordering signals generated by the user but does not teach or suggest a history of document states produced by the user. Because at least one element of the claimed invention, automatically capturing the state of a document after an operation, is not taught or suggested by Bristor, no prima facie case of obviousness under 35 U.S.C. § 103 has been established.

Claims 2 – 7 incorporate the features of claim 1 and are allowable for at least the same reasons as claim 1.

Claim 3 recites a method including a state history that has “states of the document and the order in which the stored states were automatically added to the state history” and is “displayed to a user as a list of document states shown in their stored order.” Bristor does not teach or suggest such a method. Bristor teaches the creation of a history database comprising user-generated data that may be subsequently resubmitted to a processor (col. 13, lines 30-37 and col. 17, lines 8-16). The re-execution of user data does not necessarily produce a document in the same state as when the user data was originally submitted. For example, if the web page associated with an URL has been updated, retrieving and re-executing the command to visit that page will retrieve the document in its current state rather than its previous state. Therefore, the storage of user data in the order it was generated is not the same as the storage of document states as they were generated.

The method of claim 3 also includes “performing a step backward operation” wherein “all step backward operations place the document in a state that occurred immediately after [an operation that changed the state of the document].” Bristor does not teach or suggest such a method because stepping back through and re-executing user data does not place a document in a previous state. For example, if files have been added to a directory, stepping back to a previously executed Unix shell command such as “ls mydir” and re-executing it will not produce the same list of files (nor would the files be in the same state) as when the command was previously executed. Accordingly, claim 3 is allowable over Bristor.

Claims 4 – 7 incorporate the features of claim 3 and are allowable for at least the same reasons as Claim 3.

Claim 5 recites a method that is implemented in a “digital graphics program operable to create and revise images in digital form” where “the images are raster images.” Bristor did not teach or suggest such a method. Because at least one of the elements of the claimed invention is not taught or suggested by the Bristor patent, no prima facie case of obviousness under 35 U.S.C. § 103 has been established.

Claim 6 recites a method that enables a user to select an item in the history list and “create a new document having the document state corresponding to the selected item.” Bristor did not teach or suggest such a method. Bristor taught a method that enables a user to identify easily and select a previously generated signal (col. 6, lines 29-32). The data field associated with the selected signal is then translated into a form recognized by the engine (col. 16, lines 61-67; col. 17, lines 1-7) and sent to the processor. The engine does not, however, distinguish user data generated directly by the user from user data that is reconstructed from the history database (col. 17, lines 8-20). The signal is executed as if currently inputted, acting within the current state of the system. Thus, Bristor does not teach or suggest a method for creating a new document having a selected previous state. Accordingly, claim 6 is allowable over Bristor.

Claim 8 recites a method of enabling a user to undo revisions made to a document by maintaining a first history of interesting operations and a second history of all operations requested by the user, the second history but not the first history including operations global to the state of the application. Bristor does not disclose any method for enabling a user to undo revisions made to a document. Accordingly, claim 8 is allowable over Bristor.

Bristor also does not disclose a method of maintaining a first history of interesting operations and a second history of operations global to the state of the application. Bristor teaches how to automatically generate an organizational structure for previously generated user data (col. 5, lines 62-67). The user data are generated at the user-interface and are transmitted via an engine to the processor in a computer system (col. 1, lines 13-17; col. 11, line 67, and col. 12, lines 1-13; Fig. 7). Bristor discloses that all user data received are stored in a history database or, alternatively, only user data designated by the user are stored (col. 17, lines 48-54; see also col. 5, lines 8-30). Thus, Bristor neither teaches nor suggests making any automatic

distinction among user data, let alone the distinction in operations recited in claim 8. For this reason alone, claim 8 is allowable over Bristol.

Claim 9 recites "a computer-implemented method of interacting with a user editing a document in a computer program application." In this method, the user submits "a sequence of commands to change the document" and the document state is changed "in response to each command." "[E]ach time the document state is changed," the changed document state is added "to a state history maintained in a computer-readable memory device" and "a corresponding entry to a history list [is] displayed to the user on a computer-controlled display device operated as part of a graphical user interface." "In response to a user action stepping backward to an item in the history list," the document is updated "to have the corresponding document state saved in the state history."

Bristol teaches a general method for organizing user data and using it to interface with a computer system. Bristol taught the use of the method to interact with the Unix operating system (col. 8, lines 44-50 and Figs. 1A and 1B) or to display Internet files (col. 9, lines 23-26, lines 48-51, and Figs. 3A and 3B). Bristol also teaches the application of the method to a circuit editor, where, "in response to user data, the circuit editor identifies the electrical component as selected and can redisplay circuit design" (col. 11, lines 8-60; Fig. 5A and 5B). Bristol does not teach or suggest the use of user data to change the state of a document. Bristol also does not teach or suggest adding a changed document state to a state history. For at least these reasons, claim 9 is allowable over Bristol.

Claim 9 is also allowable for at least the reasons given for claim 6.

Claims 10-15 depend from claim 9 and are allowable for at least the same reasons as claim 9.

Claim 10 recites a method where "the state history and the history list are limited to storing a preset number of items and excess items are scrolled off the top of the list as new items are added." Bristol does not teach or suggest such a method. Accordingly, claim 10 is allowable over Bristol.

Claim 11 recites a method where "the state history is stored in a region of memory and the oldest document states in the state history are discarded when free space in the region runs low." Bristol does not teach or suggest such a method. Bristol simply noted that "the particular

structure of history database depends on the nature of the use data to be stored" (col. 13, lines 33-35). Accordingly, claim 11 is allowable over Bristor.

Claim 12 recites a method where the oldest document states are found and discarded by a memory management process. Bristor does not teach or suggest such a method. Bristor simply noted that data fields "represent in memory the previously generated user data" (col. 14, lines 59-61). Accordingly, claim 12 is allowable over Bristor.

Claim 13 recites a method where "a command to change the document that comes after a step backward command to a selected item in the history list causes the items after the selected item to be deleted from the history list and the corresponding document states to be deleted from the state history." Bristor does not teach or suggest such a method. Bristor describes a method in the Netscape Web browser for avoiding the listing of redundant user data (col. 4, lines 56-65), but the Web documents that are displayed in response to the user data are not changed in response to the user data. Accordingly, claim 13 is allowable over Bristor.

Claim 14 recites a method where "a command to change the document that comes after a step backward command to a selected item in the history list does not cause the items after the selected item to be deleted from the history list and adds a new item to the end of the history list and a new document state to the state history." Applicant respectfully submits that claim 14 is allowable over Bristor for reasons given for claim 13.

Claim 15 recites a method that "enabl[es] a user interface gesture on the history list to create a new document from a document state from the state history." Bristor does not teach or suggest such a method. Bristor describes a method in web browsers for gesturing in a web document to receive the graphical representation of a second page (col. 4, lines 56-65); this gesture does not select from a history list but, rather, from Internet links in the document. Accordingly, claim 15 is allowable over Bristor.

Claim 16 recites a method that includes "keeping a history list; going back to a previous state in the history list; selecting a future state from the history list, being a state created after the previous state, as a source of data for an operation; and performing the operation with the future data on the previous state." Bristor does not teach or suggest a history list of states; rather, Bristor teaches a history database of user data (col. 7, lines 30-35), comprising user-generated signals (col. 1, lines 13-23). Bristor also does not teach or suggest performing an operation

associated with a future state on a previous state. Accordingly, claim 16 is allowable over Bristor.

Claim 17 recites a method that includes "keeping a history of document states created by a user; enabling the user to discard any of the history; and enabling the user to step backward and forward through the history and thereby to alter the state of the document to be any of the document states in the history." Bristor does not teach or suggest a history of document states; rather, Bristor teaches a history database of user data (col. 7, lines 30-35), comprising user-generated signals (col. 1, lines 13-23). Also, Bristor did not teach or suggest enabling the user to discard any of the history. Accordingly, claim 17 is allowable over Bristor.

Claim 18 recites a method that includes "keeping a history of document states created automatically whenever a user command to the application changes the state of a document; enabling the user to discard any user-selected set of the document states in the history; and enabling the user to designate any one of the document states in the history and thereby install the designated state as the current state of the document." This method is possible because the later document states are complete in themselves, such that earlier ones may be deleted with no adverse effect (specification, page 12, lines 1-3). Bristor does not teach or suggest keeping a history of document states; rather, Bristor teaches a history database of user data (col. 7, lines 30-35), comprising user-generated signals (col. 1, lines 13-23).

Bristor also does not teach or suggest enabling the user to discard any user-selected set of the document states in the history and to install the designated state as the current state of the document. Assuming that a series of user data were directed to editing a single document and making successive changes in the document state, it might be possible to recreate the earlier state using Bristor's method by, for example, undoing the effect of each user-generated signal. However, it would be impossible to designate and install the earlier state of the document as the current state if some of the intervening signals were deleted. Indeed, the claimed invention provides a way to implement multiple undo efficiently while avoiding such operation sequence dependencies (specification, page 4, lines 14-16). Accordingly, claim 16 is allowable over Bristor.

Claims 19-21 depend from claim 18 and are allowable for at least that reason.

Claim 20 recites a method where "the saved history resides in the document with final document data." Bristor does not teach or suggest saving the history with the final document data. Bristor discloses a history database of user data (col. 7, lines 30-35), comprising user-generated signals (col. 1, lines 13-23), which may or may not be associated with any particular document. Indeed, Bristor does not disclose saving a document or document data. Accordingly, claim 20 is allowable over Bristor.

Claim 22 recites a method including "identifying for the user on a display device a set of states that the document has been in by operation of the application; enabling the user to designate any one of the identified states; and providing the user an editing tool having the designated state as a document state operand." Claim 22 is allowable over Bristor for the reasons discussed above for claims 1 and 18. Bristor discloses a history database of user data (col. 7, lines 30-35), comprising user-generated signals (col. 1, lines 13-23), which may or may not be associated with any particular document. Bristor also discloses a method for designating a particular user-generated signal and executing it (e.g., col. 8, lines 44-46). Thus, Bristor did not teach or suggest identifying to the user on a display device a set of states and providing the user an editing tool having the designated state as a document state operand. Accordingly, claim 22 is allowable over Bristor.

Claims 23-25, 27-36, and 44 incorporate the features of claim 22 and are allowable for at least the same reasons. Applicant has canceled claim 26.

Claim 37 is a computer program product claim corresponding to claim 1. Claim 37 is allowable for at least the reasons set forth above in connection with claim 1.

Claim 38 is a computer program product claim corresponding to claim 8. Claim 38 is allowable for at least the reasons set forth above in reference to claim 8.

Claim 39 is allowable for the reason given for claim 9.

Claim 40 is allowable for the reason given for claim 16.

Claim 41 is allowable for the reason given for claim 17.

Claim 42 is allowable for the reason given for claim 18.

Claim 43 is a computer program product claim corresponding to claim 1 and is allowable for at least the same reasons that apply to claim 1.

Applicant : Hamburg, et al.  
Serial No. : 09/010,801  
Filed : January 22, 1998  
Page : 10

Attorney's Docket No.: 07844-235001 / P212

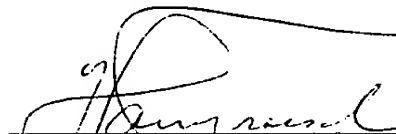
## II. Conclusions

Attached is a marked-up version of the changes being made by the current amendment.

Applicant submits that all of the claims are now in condition for allowance, which action is requested. No fee is calculated to be due with this response. However, please apply any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,

Date: 03 Jan 02



Hans R. Troesch  
Reg. No. 36,950

Fish & Richardson P.C.  
500 Arguello Street, Suite 500  
Redwood City, California 94063  
Telephone: (650) 839-5070  
Facsimile: (650) 839-5071



**Version with markings to show changes made**

**In the claims:**

Claim 26 has been canceled.

Claims 8, 22, 27, 37 and 38 have been amended as follows:

8. (Amended) A method for enabling a user to undo revisions made to a document, the method comprising:

maintaining a first history of [interesting] operations of a predetermined type and a second history of all operations requested by a user, the second history but not the first history including operations global to the state of the application.

22. A method enabling a user to control operation of a computer program application for creating and modifying a document, the method comprising:

identifying for the user on a display device a set of states that the document has been in by operation of the application;

enabling the user to designate any one of the identified states; and

providing the user an editing tool having [the designated state as] a document state operand derived from the designated state.

27. (Amended) The method of claim 22, further comprising:

providing the user a[n] delete tool for deleting the designated state from the set of states.

37. (Amended) Apparatus comprising a computer-readable storage medium tangibly embodying program instructions defining a computer program application for performing operations on documents having states, the program comprising instructions operable for causing a programmable processor to:

maintain in a memory a state history of a document for storing document states; and

whenever an [interesting] operation of a predetermined type has occurred, automatically capture the state of the document as it exists after the operation and adding the captured state to the state history.

38. (Amended) Apparatus comprising a computer-readable storage medium tangibly embodying program instructions for use by a user of a program to undo revisions made to a document, the apparatus comprising instructions operable for causing a programmable processor to:

maintain a first history of [interesting] operations of a predetermined type and a second history of all operations requested by a user, the second history but not the first history including operations global to the state of the application.